# CLAIM AMENDMENTS

## Claim Amendment Summary

**Claims pending**

- Before this Amendment: Claims 1-28, 34-42, and 45-50.
- After this Amendment: Claims 1-28, 34-42, and 45-46.

**Non-Elected, Canceled, or Withdrawn claims**: 29-33, 43, 44, and 47-50.

**Amended claims**: 1, 11-13, 26-28, 34, 38-40, and 45.

**New claims**: none.

---

## Claims:

**1.** (CURRENTLY AMENDED) A kernel emulator implemented at least in part by a computing device for non-native program modules, [[~~the kernel emulator comprising software and~~]] the kernel emulator comprising:

an interceptor configured to intercept non-native kernel calls that call a native kernel from non-native program modules, the native kernel being software that operates system functions;

a call-converter configured to convert the non-native kernel calls intercepted by the interceptor into native kernel calls; and

an I/O unit configured to deliver the native kernel calls converted by the call-converter to the native kernel.

**2.** (**ORIGINAL**) An emulator as recited in claim 1, wherein the call-converter comprises a translator configured to translate a non-native paradigm for passing parameters into a native paradigm for passing parameters.

**3.** (**ORIGINAL**) An emulator as recited in claim 1, wherein the call-converter comprises a translator configured to translate non-native CPU instructions into native CPU instructions.

**4.** (**ORIGINAL**) An emulator as recited in claim 1, wherein the call-converter comprises a translator configured to translate addresses from non-native length into native length.

**5.** (**ORIGINAL**) An emulator as recited in claim 1, wherein the call-converter comprises an argument-converter configured to convert non-native argument format into native argument format.

**6.** (**ORIGINAL**) An emulator as recited in claim 1, wherein the call-converter comprises a translator configured to translate words from non-native word size into native word size.

**7.** (**ORIGINAL**) An emulator as recited in claim 1 further comprising a memory constrainer configured to limit addressable memory to a range addressable by non-native program modules.

**8.** (**ORIGINAL**) An emulator as recited in claim 1 further comprising a shared-memory manager configured to manage memory space that is accessible to both native and non-native program modules.

**9.** (**ORIGINAL**) An emulator as recited in claim 1 further comprising a shared-memory manager configured to synchronize a native shared data structure with a non-native shared data structure.

**10.** (**PREVIOUSLY PRESENTED**) An emulator as recited in claim 1 further comprising a shared-memory manager configured to manage memory space that is accessible to both native and non-native program modules, wherein the shared-memory manager maps versions of process shared data structures (process SDSs) and versions of thread shared data structures (thread SDSs) between native and non-native program modules.

**11.** (**CURRENTLY AMENDED**) An operating system on a computer-readable medium, comprising:

a native kernel configured to receive calls from native program modules;

a kernel emulator as recited in claim 1 configured to receive and convert calls from non-native program modules, whereby the calls from the non-native program modules are processed by the native kernel through the kernel emulator without modifying the non-native program modules.

**12.** (CURRENTLY AMENDED) An operating system on a computer-readable medium, comprising:

a native kernel configured to receive calls from native APIs;

a kernel emulator as recited in claim 1 configured to receive calls from non-native APIs, whereby the calls from non-native APIs are processed by the native kernel through the kernel emulator without modifying the non-native APIs.

**13.** (CURRENTLY AMENDED) A method of emulating a kernel for non-native program modules, the method comprising:

intercepting non-native kernel calls from non-native program modules, the non-native kernel calls calling a native kernel that comprises software and operates system functions [[~~emulator comprising software~~]];

converting the intercepted non-native kernel calls into native kernel calls; and

delivering the converted native kernel calls to the native kernel, whereby the non-native kernel calls from the non-native program modules are processed by the native kernel through the conversion without modifying the non-native program modules.

**14.** (ORIGINAL) A method as recited in claim 13, wherein the converting step comprises translating a non-native paradigm for passing parameters into a native paradigm for passing parameters.

**15.** (ORIGINAL) A method as recited in claim 13, wherein the converting step comprises translating non-native CPU instructions into native CPU instructions.

**16.** (ORIGNAL) A method as recited in claim 13, wherein the converting step comprises translating addresses from non-native length into native length.

**17.** (ORIGINAL) A method as recited in claim 13, wherein the converting step comprises translating words from non-native word size into native word size.

**18.** (ORIGINAL) A method as recited in claim 13 further comprising limiting addressable memory to a range addressable by non-native program modules.

**19.** (ORIGINAL) A method as recited in claim 13 further comprising synchronizing a native shared data structure with a non-native shared data structure.

**20.** (ORIGINAL) A method as recited in claim 13 further comprising mapping versions of process shared data structures (SDSs) between native and non-native program modules.

**21.** (ORIGINAL) A method as recited in claim 20, wherein a process SDS of a native program module includes a pointer to a process SDS of a non-native program module.

**22.** (ORIGINAL) A method as recited in claim 20, wherein a process SDS of a non-native program module includes a pointer to a process SDS of a native program module.

**23.** (ORIGINAL) A method as recited in claim 13 further comprising mapping versions of thread shared data structures (SDSs) data structure between native and non-native program modules.

**24.** (ORIGINAL) A method as recited in claim 23, wherein a thread SDS of a native program module includes a pointer to a thread SDS of a non-native program module.

**25.** (ORIGINAL) A method as recited in claim 23, wherein a thread SDS of a non-native program module includes a pointer to a thread SDS of a native program module.

**26.** (CURRENTLY AMENDED) A computer comprising one or more computer-readable media having computer-executable instructions that, when executed by the computer, perform the method as recited in claim 13, whereby the non-native kernel calls from the non-native program modules are processed by the native kernel through the conversion without modifying the non-native program modules.

**27.** (CURRENTLY AMENDED) A computer-readable medium having computer-executable instructions that, when executed by a computer, performs the method as recited in claim 13, whereby the non-native kernel calls from the non-native program modules are processed by the native kernel through the conversion without modifying the non-native program modules.

**28.** (CURRENTLY AMENDED) An operating system embodied on a computer-readable medium having computer-executable instructions that, when executed by a computer, performs the method as recited in claim 13, whereby the non-native kernel calls from the non-native program modules are processed by the native kernel through the conversion without modifying the non-native program modules.

**29-33** (CANCELED).

**34.** (CURRENTLY AMENDED) A method comprising emulating a non-native kernel for a native computing platform so that non-native kernel calls that call a native kernel from non-native applications are [[translated]] converted into native kernel calls to [[a]] the native kernel, the native kernel [[~~emulator comprising software~~]] comprising software that operates system functions.

**35.** (ORIGINAL) A method as recited in claim 34, wherein the emulating step comprises:

translating non-native CPU instructions into native CPU instructions;

translating addresses from non-native length into native length;

limiting addressable memory to a range addressable by non-native program modules.

**36.** (ORIGINAL) A method as recited in claim 35, wherein the emulating step further comprises translating a non-native paradigm for passing parameters into a native paradigm for passing parameters.

**37.** (ORIGINAL) A method as recited in claim 34, wherein the converting step further comprises translating words from non-native word size into native word size.

**38.** (CURRENTLY AMENDED) A computer comprising one or more computer-readable media having computer-executable instructions that, when executed by the computer, perform the method as recited in claim 34, whereby the non-native kernel calls from the non-native program modules are processed by the native kernel through the conversion without modifying the non-native program modules.

**39.** (CURRENTLY AMENDED) A computer-readable medium having computer-executable instructions that, when executed by a computer, performs the method as recited in claim 34, whereby the non-native kernel calls from the non-native program modules are processed by the native kernel through the conversion without modifying the non-native program modules.

**40.** (CURRENTLY AMENDED) A kernel emulator implemented at least in part by a computing device [[configured]] to emulate a non-native kernel for a native computing platform so that non-native kernel calls that call a native kernel from non-native applications are [[translated]] converted into native kernel calls to [[a]] the native kernel, [[~~the kernel emulator comprising software~~]] the native kernel comprising software that operates system functions, whereby the non-native kernel calls from the non-native program modules are processed by the native kernel through the conversion without modifying the non-native applications.

**41.** (ORIGINAL) An emulator as recited in claim 40, wherein the emulator comprises:

an instruction-translator configured to translate non-native CPU instructions into native CPU instructions;

an address-translator configured to translate addresses from non-native length into native length;

an memory constrainer configured to limit addressable memory to a range addressable by non-native program modules.

**42.** (PREVIOUSLY PRESENTED) An operating system on a computer-readable medium, comprising:

a native kernel configured to receive calls from native program modules;

a kernel emulator as recited in claim 40 configured to receive calls from non-native program modules.

**43.** (CANCELED).

**44.** (CANCELED).

**45.** (CURRENTLY AMENDED) A kernel emulator implemented at least in part by a computing device for non-native program modules, the kernel emulator comprising software and the kernel emulator comprising:

an interceptor configured to intercept non-native kernel calls that call a native kernel from non-native program modules, the native kernel being software that operates system functions;

a call-converter configured to convert the non-native kernel calls intercepted by the interceptor into native kernel calls, wherein the call-converter comprises:

an instruction-translator configured to translate non-native CPU instructions into native CPU instructions;

an address-translator configured to translate addresses from non-native length into native length; and

an I/O unit configured to deliver converted native kernel calls to the native kernel.

**46.** (ORIGINAL) An operating system on a computer-readable medium, comprising:

a native kernel configured to receive calls from native program modules;

a kernel emulator as recited in claim 45 configured to receive calls from non-native program modules.

**47-50.** (CANCELLED).